JUN 3 0 2006

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	28.Jun.06	MAJOR REPORT

4. TITLE AND SUBTITLE
SOFTWARE QUALITY MANAGEMENT RECOMMENDATIONS.

5. FUNDING NUMBERS

6. AUTHOR(S)
MAJ HERVEY MARCUS W

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UNIVERSITY OF HOUSTON

8. PERFORMING ORGANIZATION REPORT NUMBER
CI04-1809

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

20060706090

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS	15. NUMBER OF PAGES	16. PRICE CODE
	15	

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Software Quality Management Recommendations

Publication: Crosstalk: The Journal of Defense Software Engineering
Theme: Management Basics
Submittal Deadline: 19 Jun 06
Publishing Date: November 06

Abstract:

Quality has always been an important element of successful software development and maintenance. However, quality seems to always take a back seat to cost and schedule. Delivering deficient products on time and under budget may still result in project failure. Military organizations responsible for the acquisition and development of software must focus on quality in order to keep up with the increased demands for complex, software-intensive systems. This article presents some of the major components of software quality management and offers ten recommendations to improve software quality within your organization.

THE VIEWS EXPRESSED IN THIS ARTICLE ARE THOSE OF THE AUTHOR AND DO NOT REFLECT THE OFFICIAL POLICY OR POSITION OF THE UNITED STATES AIR FORCE, DEPARTMENT OF DEFENSE, OR THE U.S. GOVERNMENT.

Article:

A management discussion on software-intensive systems would not be complete without highlighting the importance of effective software quality management. 30 years after Fred Brooks' ground-breaking book, The Mythical Man Month, consistent software quality continues to elude the software development community. A major cause is the continued lack of focus on quality throughout many software development organizations and software projects. In many cases, the focus of program/project management is primarily targeted on cost and schedule, while sacrificing performance. An oversight of quality issues can cause problems to remain hidden until after the product has been released, therefore being more costly to resolve. Most quality problems result from an immature process, an outdated process or a lack of discipline to follow a process. Military organizations responsible for the acquisition and development of software must continue to institutionalize and implement effective software quality management techniques, to keep up with the increased demands for complex, software-intensive systems. This article presents some of the major components of software quality management and offers ten recommendations to improve software quality within your institution.

To understand the value of the proposed recommendations, it is important to dissect the goal of software quality into some of its major components. In this article, we will discuss the five components of leadership, planning, process management, project management, and education and training. These five components are shown in Figure 1 and their relationships are shown in Figure 2.

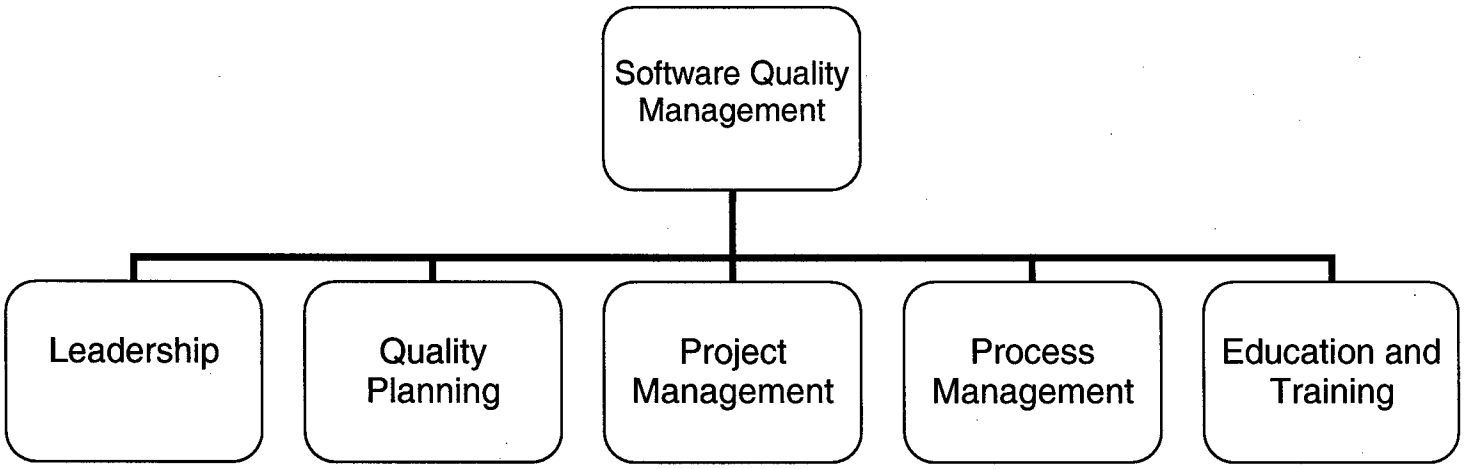


Figure 1. Major Components of Organizational Software Quality

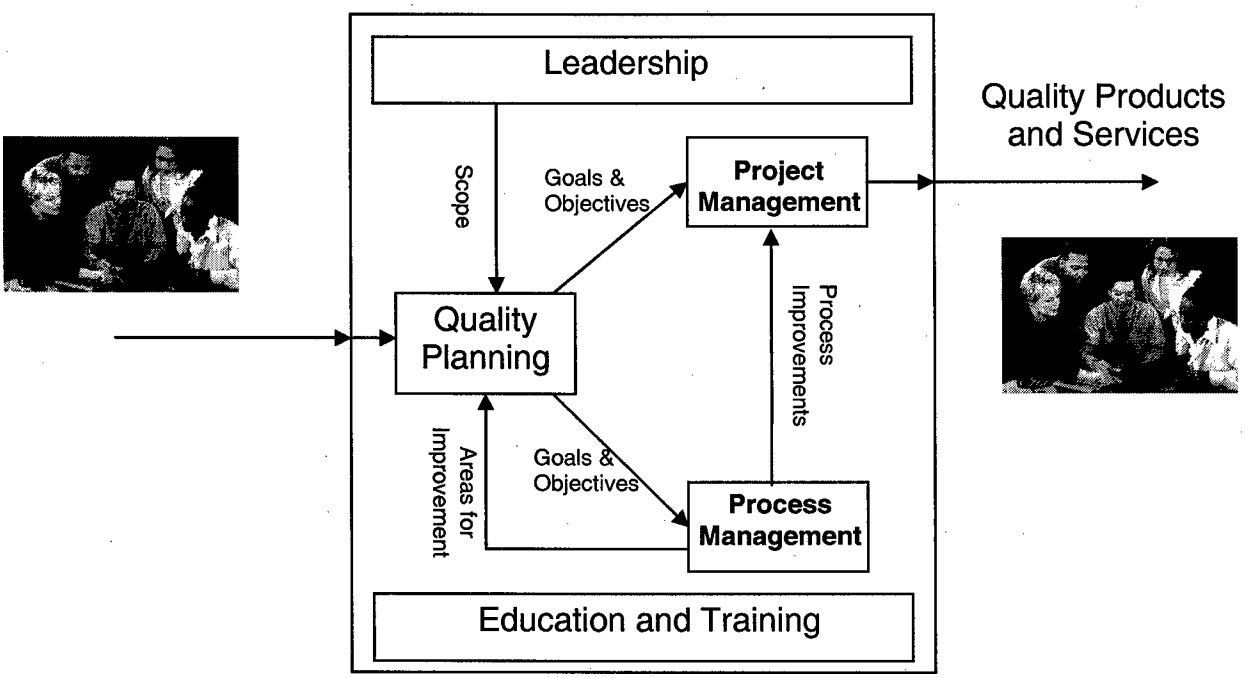


Figure 2. Relationship between Major Software Quality Components

Leadership

Leadership, the first component, is fundamental for software quality efforts to succeed. Senior management, within an organization, must make a visible commitment to quality, and must inspire others in the organization to follow defined processes and implement software quality best practices.

Leadership is also needed to ensure quality principles are institutionalized. This can be achieved through the development of an environment that enables a mindset in which quality is the responsibility of everyone, and each individual feels like a contributing member of a team with an investment in the final product.

Quality Planning

Once leadership has provided the scope, quality planning naturally follows as the second major component. Software quality efforts have to be planned in order to be successful, and this component involves identifying relevant and realistic quality goals and objectives that can be measured.

Project Management

Project management, the third component, is essential to delivering high quality software systems. While all aspects of project management are important, this article highlights the importance of risk and configuration management. Risk management involves identifying and tracking quality and performance risks, and implementing

appropriate mitigation strategies as early as possible. Configuration management is also important and enables change and error control throughout the software lifecycle.

Process Management

Just as specific aspects of a project, need to be managed, the same is also true for organizational software processes. The fourth component of software quality is process management and involves ensuring new processes are defined, followed and updated.

A key part of process management involves continuous process improvement. Every process can be enhanced. The challenge is identifying where improvements should be made and using change management techniques to effectively implement the process improvements.

Quality Education and Training

The final component is education and training. To be effective, personnel have to be trained on many aspects of software quality. Knowledge, skills and abilities must be taught and demonstrated on software quality principles, the organization's software development process, the importance of software metrics, and software quality assessments models. Personnel must also be made aware of how they can improve their own personal processes.

RECOMMENDATIONS

The previous review of some major software quality components provides a foundation for the following recommendations proposed. These recommendations should not replace a software quality program specifically tailored for your project or organization, but could be used as a catalyst for your organization-wide software quality initiatives. Table 1 lists ten software quality improvement recommendations and their relation to the software quality components previously discussed.

Recommendation	Component Area
1. Focus on a common **software "quality" definition**	Leadership
2. Focus on **software quality planning**	Quality Planning
3. Focus on developing **"quality" people**	Education & Training
4. Focus on **quality assessments**	Process Management
5. Focus on **requirements**	Project Management
6. Focus on creating an effective **SQA group**	Process Management
7. Focus on **risk mitigation**	Project Management
8. Focus on **defect prevention**	Process Management
9. Focus on **software quality metrics**	Project/Process Management
10. Focus on **teamwork**	Project Management

Table 1. 10 Quality Improvement Recommendations

Recommendation 1: Focus on a common software "quality" definition

What does quality mean to your organization? One of the first steps in developing a software quality program is to come to a consensus on the definition of quality within your organization and with your users and customers. This definition can be used to help communicate the organization's vision for software quality and is commonly documented in an organizational software quality policy.

Microsoft has improved their understanding of what quality means to its customers and has renewed its focus on software quality in the development of the anticipated Windows Vista operating system. A company spokesperson recently stated that "...the top priority for Windows Vista is quality." This focus on quality specifically led to a mature decision to remove a highly anticipated synchronization feature from the Windows Vista operating system because it was not performing to a customer acceptable quality level.

Recommendation 2: Focus on software quality planning

Software quality efforts have to be planned in order to be successful. Documenting plans also provides a mechanism to monitor and compare planned with actual results. Two specific plans that are useful in documenting the results of the software quality planning process include the Software Quality Assurance Plan (SQAP) and the Software Verification and Validation Plans (SVVP).

The SQAP is used to document quality objectives and identify milestones with respect to product and process quality initiatives. The SVVP is used to document the specific activities your organization will perform to verify and validate the quality of the software product. Templates for the contents of a SQAP and a SVVP can be found in IEEE Standard for Software Quality Assurance Plans (IEEE Std 730-1998) and IEEE Standard for Software Verification and Validation Plans (IEEE Std 1059-1993), respectively. However, the templates should be tailored to represent your specific development efforts.

Recommendation 3: Focus on people

People are an organization's most important resource. Therefore it's important for any software development organization to make an investment in ensuring its personnel understand the importance of quality, are trained in the organization's development process, and know how to implement recommended software quality best practices. Specifically, development of personnel can be achieved by providing education and training programs on quality within your organization and through the encouragement of individuals to obtain professional quality and software engineering training and/or obtain certifications.

Three professional certification programs are highlighted that may be beneficial in improving the skills of personnel toward building quality software systems. The three certification programs include the Certified Software Quality Engineer (CSQE), the Certified Software Development Professional (CSDP) and the Project Management Professional (PMP) certifications.

The American Society for Quality (ASQ) offers many certifications related to quality. Specifically, the CSQE certification is most appropriate for individuals interested in software quality. This certification tests proficiency in 8 knowledge areas related to software quality fundamentals.

The Institute for Electrical and Electronic Engineers (IEEE) Computer Society offers the Certified Software Development Professional (CSDP) certification. This certification tests ten knowledge areas based upon the Software Engineering Body of Knowledge (SWEBOK).

The Professional Management Institute (PMI) offers the Project Management Professional (PMP) certification that certifies proficiency in 9 knowledge areas based upon the Project Management Body of Knowledge (PMBOK).

The Air Force Institute of Technology School of Systems and Logistics at Wright-Patterson Air Force Base also offers the Software Professional Development Program (SPDP) which is provided free to all Department of Defense employees. This program offers a distance learning format that can provide a more practical software engineering education. SPDP is also a registered education provider for the IEEE CSDP certification program. This allows individuals who complete SPDP courses to earn points towards fulfilling CSDP recertification requirements. More information on the AFIT SPDP program can be found at http://ls.afit.edu.

Recommendation 4: Focus on software quality assessments

How do you know how far you have to go to reach your goal, if you don't know where you are? Assessments are important because they provide a basis for your quality improvement program and highlight strengths and identify weaknesses. Software quality process assessments are useful in providing software organizations an assessment of their software process maturity. The Software Engineering Institute's (SEI) Capability Maturity Model Integration (CMMI) and the International Standards Organization (ISO) 9001 are two software quality assessment models that can be used to assess the maturity level of your organization. The CMMI provides a both a continuous and a stage representation model for software development maturity.

Recommendation 5: Focus on requirements

A quality system results from a focus on requirements that are traceable and can be validated. It is important to spend time early in the software development lifecycle to create quality requirements that can be traced backwards to system requirements and traced forward to design specifications, test specifications, test procedures, test cases, and test results. It is equally important to understand how functional requirements will be validated as they are defined in the software requirements specification. Common validation methods include demonstration, test, analysis or inspection. Identifying the validation method of a requirement at during its definition also enables better requirements to be created. Effective communication is the key in defining "quality" requirements. Table 2 displays a list of some common attributes that quality requirements should possess.

Software Quality Attributes	
• Accuracy	• Portability
• Completeness	• Readability
• Consistency	• Reusability
• Correctness	• Reliability
• Efficiency	• Safety
• Expandability	• Security
• Flexibility	• Survivability
• Interoperability	• Testability
• Maintainability	• Usability
• Manageability	

Table 2. Software Quality Attributes

Recommendation 6: Focus on creating an effective SQA group

An effective Software Quality Assurance (SQA) group is an important element of ensuring software quality. The main goal of SQA is to provide senior management with information pertaining to the quality of the product and process. This is accomplished throughout the software lifecycle and includes oversight over software quality related functions, auditing of the product and process and reporting quality concerns. SQA is a major function and should be staffed with experienced personnel that are proficient in quality assurance techniques and able to contribute to projects in a fashion that is seen as being a positive influence in the development, instead of a negative one.

Recommendation 7: Focus on risk mitigation

Every project has risks that can impede the progress or quality of a software product. Therefore it is important to seriously identify any potential risks that may occur. Once risks have been identified and categorized based on their severity and probability of occurrence, potential risk mitigation strategies should be developed. The earlier risks can be identified, monitored and mitigated, the better the chance the project will not be negatively affected.

Recommendation 8: Focus on defect prevention

Defect prevention is more important that defect detection. Defect detection is a reactive role, where defect prevention is a proactive role. An IBM innovation, called the Defect Prevention Process, uses a closed-loop method to detect errors, analyze them and change the process to prevent them from occurring again. The idea is that you

make an error once, and then learn from that error in order to prevent ever making the same error again. This form of defect prevention works well if an organization analyzes their work after it is completed it was integrated into the development process. One disadvantage of this approach is to become overwhelmed with the number of potential improvements, but this be remedied through prioritization.

Recommendation 9: Focus on Software Quality Metrics

The identification of appropriate performance measures is important in managing the quality of a software-intensive system. Metrics allow the project to be quantitatively monitored and can be useful in identifying trends and problem areas. It is important to consciously determine the right metrics to use to help ensure quality. They should be meaningful and attainable, based on how the data is gathered. Appropriate metrics should also be identified to monitor both the quality of the product and the process. Some examples of product and process metrics can be found in Table 3.

Software Product Metrics	Software Process Metrics
• Percentage of requirements demonstrated during testing	• Number of requirements successfully traced through design, code and test
• Number of known defects	• Number of defects introduced per developer hour
• Ratio of tests passed to tests performed	• Number of items reworked

Table 3. Software Product and Process Metrics

Recommendation 10: Focus on teamwork

Software development is a team sport. Successful teams can accomplish major goals and conquer major challenges. Effective teamwork is even more necessary with the increased need for complex software systems. Everyone must know their part and roles and responsibilities should be well defined and understood. The key to teamwork is effective communication about the project and the process. Creating or updating your organizational software process handbook, coding standards and developing naming conventions. These products are effective ways to enhance team communication and foster better teamwork. The development of a best practices or lessons learned database can may also be beneficial in improving team performance through learning from past mistakes.

CONCLUSIONS

Reputations are built around quality, and software quality management must be looked at as more than just a necessary evil, but as a guiding principle. This article presented some common software quality techniques that will allow software managers and engineers to build higher quality software systems. Implementing a few best practices can improve the confidence in your organization's products and processes. The ten recommended actions can be used to start improving the management of software quality within your organization or as a start to developing a more robust software quality program.

The investment needed to achieve software quality requires effort, but the benefits of meeting customer and end user needs are well worth it. Most software

quality concepts are not difficult to understand, however organizational management and software personnel must practice discipline in adhering to these concepts and be willing to continually improve their processes. So while most program/project management may currently focus on cost and schedule, we offer ten recommendations why you may want to try a new prescription. **FOCUS ON QUALITY!**

REFERENCES

[1] Brooks, Frederick. The Mythical-Man Month: Essays on Software Engineering. New York: Addison Wesley Longman, Inc., 1995.

[2] Davis, Alan M. Software Requirements. Upper Saddle River: Prentice Hall PTR, 1993.

[3] Foley, Mary Jo. "Another Windows Vista Feature Bites the Dust." Microsoft Watch. http:www.microsoft-watch.com, June 7, 2006.

[4] Humphrey, Watts. Managing the Software Process. New York: Addison-Wesley Publishing Company, Inc., 1995.

[5] IEEE Std 730-1998. IEEE Standard for Software Quality Assurance Plans.

[6] IEEE Std 829-1998. IEEE Standard for Software Test Documentation.

[7] IEEE Std 830-1988. IEEE Recommended Practice for Software Requirements Specifications.

[8] IEEE Std 982.2-1988. IEEE Guide for Use of IEEE Standard Dictionary of Measures to Produce Reliable Software.

[9] IEEE Std 1016-1998. IEEE Recommended Practice for Software Design Descriptions.

[10] ANSI/IEEE Std 1042-1987. IEEE Guide to Software Configuration Management.

[11] IEEE Std 1044.1-1995. IEEE Guide to Classification for Software Anomalies.

[12] IEEE Std 1059-1993. IEEE Guide for Software Verification and Validation Plans.

[13] IEEE Std 1061-1998. IEEE Standard for a Software Quality Metrics Methodology.

[14] MIL-HDBK-286. A Guide for DOD-STD-2168, Defense System Software Quality Program.

[15] MIL-Q-9858A. Quality Program Requirements.

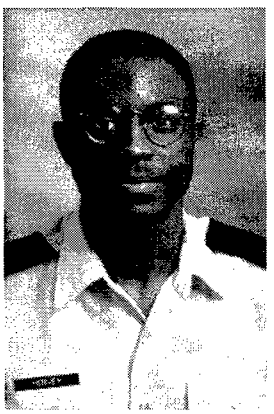
[16] IEEE Std 1059-1993. IEEE Standard for Software Verification and Validation Plans

[17] IEEE Std 1061-1998. IEEE Standard for a Software Quality Metrics Methodology.

[18] Kaplan, Craig, Ralph Clark, and Victor Tang. Secrets of Software Quality. New York: McGraw-Hill, Inc., 1995.

[19] Pressman, Roger S. Software Engineering: A Practical Approach. 3rd ed. New York: McGraw-Hill, Inc., 1992.

[20] Yourdan, Edward. Decline & Fall of the American Programmer. Englewood Cliffs: PTR Prentice-Hall, Inc., 1992.



Major Marcus W. Hervey, PMP, is currently a student at the University of Houston, working towards a doctorate degree in Computer Science, and aspiring to become an instructor for the Software Professional Development Program at the Air Force Institute of Technology. He has more than 14 years experience in the U.S. Air Force with over 10 years experience in systems engineering and software development and management. He holds a Bachelor of Science degree in Electrical Engineering from the United States Air Force Academy and a Master of Science degree in Computer Information Systems from St. Mary's University in San Antonio, Texas. He is a member of the American Society of Quality, the Institute of Electrical and Electronic Engineers and the Project Management Institute.

Air Force ROTC Det 003
Garrison Rm 109F
University of Houston
Houston, TX 77204-5048
(713) 429-4091
E-mail: mwhervey@cs.uh.edu